The case for extensible operating systems for exascale

Rolf Riesen, IBM Research* Kurt Ferreira, Sandia National Laboratories

Position statement

We predict that, initially, system software for exascale systems will be rapidly evolving. New technologies will become available and different ways of using them will come and go. New types of applications and programming models will also introduce new requirements. Features traditionally provided by operating systems (OSes) may not be needed in some cases, while new mechanisms, such as dealing with a much more complex memory hierarchy and the ability to work around faults or operate in degraded mode, need to be added.

General OS research will not address exascale-specific challenges since their focus is on consumer OSes and data centers [16]. Also, these OSes are not always suitable for HPC [5], and wont be for exascale. Issues such as OS-noise [24, 15], fault tolerance, and scalability need to be specifically addressed. Furthermore, an exascale OS must enable novel methods of using the hardware in application-specific ways e.g., [6]. Mainstream OS researchers working on parallel systems, often aim their efforts at MapReduce [8, 9] and Hadoop [1], which are only one type of parallel applications and not representative of the range of application types and algorithms expected at exascale.

We propose extremely lightweight kernels that can be extended by the runtime system (trusted) or the applications themselves (not trusted). This will result in highly efficient system software tailored to a given architecture, the current usage model, and the application(s) running at the moment. While some of the research necessary to accomplish this has been done in the past, a lot remains to be done to make this a reality.

Running untrusted user-level code inside an OS kernel has been studied in the 1990's but has not really caught on. We believe the time has come to resurrect kernel extensions and combine them with lightweight kernels for OSes that run on highly-parallel systems. Exascale system kernels must be able to quickly adapt to the type of application currently running as well as adapt to the diverse and evolving types of systems. They must do this without sacrificing performance or scalability.

Lightweight kernels [34, 20, 27, 22, 23] have proved themselves running highly parallel systems. Due to their smallness, they are relatively easy to expand. However, a generalized method is needed to allow customization for a specific environment and allow applications to tailor kernel behavior at runtime; e.g., to set scheduling memory access policy. While counterintuitive, small sections of interpreted code can provide the performance, flexibility, and protection necessary for kernel extensions [14, 18, 10].

In [11] the authors wonder whether extensible systems lead us astray. Again, their points are mainly directed at general purposes OSes and do not address the need of OSes specific to exascale. In [17] the Exokernel designers make many of the points for kernel extensions we make. Given the specialized environment and usage of a massively parallel system, we need even fewer of the mechanisms an Exokernel provides. Going with the philosophy of removing as much code and mechanism from the kernel as possible, we are proposing a kernel that is even more lightweight than the Exokernel. However, such a kernel needs to be extensible by the runtime system to tailor it to a specific machine and usage environment. Applications also need to be able to extend the OS for their specific needs. The latter needs to be done in a protected manner.

The ideas presented in this proposal are explored in more detail in [28].

*corresponding author: rolf.riesen@ie.ibm.com

Related work

The idea of executing user code, an extension, inside the kernel has seen several incarnations. Software-based fault isolation [33], is the idea of limiting data accesses to a certain segment of main memory. SPIN [3] used a trusted compiler to generate digitally signed spindles that get inserted into the kernel. Global Layer Unix (GLUnix) [31] used software-based fault isolation to move OS functionality into user level libraries. We want to move user code functionality into the kernel. The VINO kernel was designed to let applications specify the policies the kernel uses to manage resources. That is what we are interested in, but specifically for high-performance parallel environments, instead of database management systems for which VINO was designed.

In the μChoices OS [7, 30] agents can be inserted into the kernel. These agents are written in a simple, flexible scripting language similar to TCL, and are interpreted. These agents are mostly simple optimizations to eliminate the overhead of mutiple system calls. The MIT Exo kernel [13, 12] is similar to the approach proposed here, but the concept needs to be extended to push some functionality back into the kernel when it is needed at run time. Methods to safely execute untrusted code in a privileged environment are compared in [29]. Corey's [4] model of work assignment and sharing data among cores are some of the things our extensions are supposed to do. However, we believe that these decisions are better made at the application level. OS support is needed to implement policy.

Interpreters have been studied extensively and some of them have been embedded in kernels before. The BSD packet filter is one example [21]. Another is our work of adding a FORTH interpreter to the firmware of a Myrinet network interface [32]. Interpreters are often considered to be too slow for system services. However, our extensions are small and perform simple tasks. Techniques to execute them efficiently exist [25, 26]. Then, using direct or indirect threaded code techniques, extremely fast interpreters can be built [2, 10, 19, 14].

Assessment

Making lightweight kernels extensible will allow OSes to scale and adapt quickly to new requirements of exascale systems and computing. What kind of extensibility is needed and how to make it best accessible to applications and system software developers is an open research topic.

Challenges addressed: Exascale systems will be diverse in architecture, usage, and the applications they run. They will also be evolving and, initially, require frequent and quick modifications to system software. There will also be a need to easily try out new systems research ideas.

Maturity: Extensible OSes have been enabling research and experiments for many years. Lightweight kernels have been in production use on some of the largest systems in the world.

Uniqueness: Extensible, lightweight OSes have not caught on in the mass market place and receive little research attention. However, consumer OSes are not flexible enough and do not scale as required. A 10% performance loss in a $200 million machine is a significant loss of investment.

Novelty: Most of the focus is on expanding consumer OSes with little regard to the specific needs of the HPC community. Lightweight kernels currently in use on large-scale machines are not extensible at runtime or by the applications using them.

Applicability: In principle, with a large enough variety of powerful extensions, an OS can be configured for many different architectures and uses. Specialized configurations can be used for parallel, floating-point heavy computations, and others for more data-centric operations.

Effort: Defining, implementing, and agreeing on a good set of mechanisms and APIs will take a handful of years to be mature enough to be used routinely. Several small teams are needed in a collaborative effort. Involvement of application developers interested in cutting edge systems is also required to make this a success.

References

[1] Apache Hadoop Project Members. Welcome to apache hadoop! `http://hadoop.apache.org`, April 2009.

[2] James R. Bell. Threaded code. *Communications of the ACM*, 16(6):370–372, June 1973.

[3] Brian N. Bershad, Craig Chambers, Susan Eggers, Chris Maeda, Dylan McNamee, Przemyslaw Pardyak, Stefan Savage, and Emin Gün Sirer. SPIN - an extensible microkernel for application-specific operating system services. Technical Report CSE-94-03-03, University of Washington, February 1994.

[4] Silas Boyd-Wickizer, Haibo Chen, Rong Chen, Yandong Mao, Frans Kaashoek, Robert Morris, Aleksey Pesterev, Lex Stein, Ming Wu, Yuehua Dai, Yang Zhang, and Zheng Zhang. Corey: An operating system for many cores. In *OSDI'08: Proceedings of the 8th conference on Symposium on Operating Systems Design & Implementation*, Berkeley, CA, USA, December 2008. USENIX Association.

[5] Ron Brightwell, Arthur B. Maccabe, and Rolf Riesen. On the appropriateness of commodity operating systems for large-scale, balanced computing systems. In *International Parallel and Distributed Processing Symposium (IPDPS '03)*, April 2003.

[6] Ron Brightwell, Kevin Pedretti, and Trammell Hudson. Smartmap: operating system support for efficient data sharing among processes on a multi-core processor. In *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, 2008.

[7] Roy H. Campbell and See-Mong Tan. μChoices: An object-oriented multimedia operating system. In *Fifth Workshop on Hot Topics in Operating Systems (HotOS V)*, pages 90–94, May 1995.

[8] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. In *OSDI'04: Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation*, pages 10–10, Berkeley, CA, USA, 2004. USENIX Association.

[9] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008.

[10] Robert B. K. Dewar. Indirect threaded code. *Communications of the ACM*, 18(6):330–331, June 1975.

[11] P. Druschel, V.S. Pai, and W. Zwaenepoel. Extensible kernels are leading os research astray. pages 38–42, May 1997.

[12] Dawson R. Engler and M. Frans Kaashoek. Exterminate all operating system abstractions. In *Proceedings of HotOS V*, May 1995.

[13] Dawson R. Engler, M. Frans Kaashoek, and James O'Toole, Jr. Exokernel: An operating system architecture for application-level resource management. In *Proceedings of the Fifteenth ACM Symposium on Operating Systems Principles*, pages 251–266, December 1995.

[14] M. Anton Ertl. A portable Forth engine. In *EuroFORTH '93 conference proceedings*, 1993.

[15] Kurt B. Ferreira, Ron Brightwell, and Patrick G. Bridges. Characterizing application sensitivity to OS interference using kernel-level noise injection. In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing (Supercomputing'08)*, November 2008.

[16] Galen C. Hunt, James R. Larus, David Tarditi, and Ted Wobber. Broad new OS research: Challenges and opportunities. In *Proceedings of the 10th conference on Hot Topics in Operating Systems (HotOS IV)*, 2005.

[17] M. Frans Kaashoek, Dawson R. Engler, Gregory R. Ganger, Hector M. Briceño, Russell Hunt, David Mazières, Thomas Pinckney, Robert Grimm, John Jannotti, and Kenneth Mackenzie. Application performance and flexibility on exokernel systems. In *SOSP '97: Proceedings of the sixteenth ACM symposium on Operating systems principles*, pages 52–65, New York, NY, USA, 1997. ACM.

[18] Paul Klint. Interpretation techniques. *Software Practice and Experience*, 11:963–973, 1979.

[19] Peter M. Kogge. An architectural trail to threaded-code systems. *IEEE Computer*, 15(3):22–32, March 1982.

[20] Arthur B. Maccabe, Patrick G. Bridges, Ron Brightwell, Rolf Riesen, and Trammell Hudson. Highly configurable operating systems for ultrascale systems. In *First International Workshop on Operating Systems, Programming Environments and Management Tools for High-Performance Computing on Clusters (COSET-1)*, pages 33–40, June 2004.

[21] Jeffrey C. Mogul, Richard F. Rashid, and Michael J. Accetta. The packet filter: An efficient mechanism for user-level network code. In *Proceedings of the Eleventh ACM Symposium on Operating Systems Principles*, pages 39–51, 1987.

[22] J. E. Moreira, G. Almási, C. Archer, R. Bellofatto, P. Bergner, J. R. Brunheroto, M. Brutman, J. G. Castaños, P. G. Crumley, M. Gupta, T. Inglett, D. Lieber, D. Limpert, P. McCarthy, M. Megerian, M. Mendell, M. Mundy, D. Reed, R. K. Sahoo, A. Sanomiya, R. Shok, B. Smith, and G. G. Stewart. Blue Gene/L programming and operating environment. *IBM J. Res. Dev.*, 49:367–376, March 2005.

[23] José E. Moreira, Valentina Salapura, George Almasi, Charles Archer, Ralph Bellofatto, Peter Bergner, Randy Bickford, Mathias Blumrich, José R. Brunheroto, Arthur A. Bright, Michael Brutman, José G. Castaños, Dong Chen, Paul Coteus, Paul Crumley, Sam Ellis, Thomas Engelsiepen, Alan Gara, Mark Giampapa, Tom Gooding, Shawn Hall, Ruud A. Haring, Roger Haskin, Philip Heidelberger, Dirk Hoenicke, Todd Inglett, Gerrard V. Kopcsay, Derek Lieber, David Limpert, Pat McCarthy, Mark Megerian, Mike Mundy, Martin Ohmacht, Jeff Parker, Rick A. Rand, Don Reed, Ramendra Sahoo, Alda Sanomiya, Richard Shok, Brian Smith, Gordon G. Stewart, Todd Takken, Pavlos Vranas, Brian Wallenfelt, Blockso Michael, and Joe Ratterman. The Blue Gene/L supercomputer: a hardware and software story. *Int. J. Parallel Program.*, 35:181–206, June 2007.

[24] Fabrizio Petrini, Darren Kerbyson, and Scott Pakin. The case of the missing supercomputer performance: Achieving optimal performance on the 8,192 processors of ASCI Q. In *SC '03: Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, 2003.

[25] Thomas Pittman. Two-level hybrid interpreter/native code execution for combined space-time program efficiency. In *Proceedings of the SIGPLAN'87 Symposium on Interpreters and Interpretive Techniques*, pages 150–152, June 1987.

[26] Todd A. Proebsting. Optimizing an ANSI C interpreter with superoperators. In *22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 322–332, January 1995.

[27] Rolf Riesen, Ron Brightwell, Patrick G. Bridges, Trammell Hudson, Arthur B. Maccabe, Patrick M. Widener, and Kurt Ferreira. Designing and implementing lightweight kernels for capability computing. *Concurrency and Computation: Practice and Experience*, 21(6):793–817, April 2009.

[28] Rolf Riesen and Kurt Ferreira. An extensible operating system design for large-scale parallel machines. Technical report SAND09-2660, Sandia National Laboratories, April 2009.

[29] Christopher Small and Margo Seltzer. A comparison of OS extension technologies. In *1996 USENIX Annual Technical Conference*, January 1996.

[30] See-Mong Tan, David K. Raila, and Roy H. Campbell. An object-oriented nano-kernel for operating system hardware support. In *Proceedings of the Fourth International Workshop on Object Orientations in Operating Systems*, pages 220–223, August 1995.

[31] Amin Vahdat, Douglas Ghormley, and Thomas Anderson. Efficient, portable, and robust extension of operating system functionality. Technical Report UCB CS-94-842, Computer Science Division, UC Berkeley, December 1994.

[32] Adam Wagner, Hyun-Wook Jin, Dhabaleswar K. Panda, and Rolf Riesen. NIC-based offload of dynamic user-defined modules for Myrinet clusters. In *IEEE Cluster Computing*, pages 205–214, September 2004.

[33] Robert Wahbe, Steven Lucco, Thomas E. Anderson, and Susan L. Graham. Efficient software-based fault isolation. In *Proceedings of the Fourteenth ACM Symposium on Operating Systems Principles*, pages 203–216, December 1993.

[34] Stephen R. Wheat, Arthur B. Maccabe, Rolf Riesen, David W. van Dresser, and T. Mack Stallcup. PUMA: An operating system for massively parallel systems. *Scientific Programming*, 3:275–288, 1994.